Analyzing using Software Defined Radios as Wireless Sensor Network Inspection and Testing Devices: An Internet of Things Penetration Testing Perspective

George D. O'Mahony
Dept. of Electrical and Electronic Engineering, University College Cork
Cork, Ireland
george.omahony@umail.ucc.ie

Philip J. Harris
Raytheon Technologies Research Center
Cork,
Ireland
harrispj@rtx.com

Colin C. Murphy
Dept. of Electrical and Electronic Engineering, University College Cork
Cork, Ireland
colinmurphy@ucc.ie

***Abstract*—Wireless sensor network (WSN) research and development is producing viable solutions for various innovative applications, including critical areas such as the Internet of Things (IoT), which is becoming a significant feature of modern technology. WSNs form an integral component of the IoT infrastructure by, frequently, implementing the communication links between sensors and the access point or central coordinator. This design and use in IoT applications intensifies the incentive to attack WSNs as sensitive data is available and transmitted in wireless links, which inherently contain security vulnerabilities, especially from external malicious interference. To ensure satisfactory performance, safety and privacy, communication links and WSN devices must be secure. Hence, penetration testing to identify security vulnerabilities and responses to external intrusions is a prerequisite to forming secure connections and an overall secure network. Derived from a prior study, this paper explores the benefits of using software-defined radios (SDRs) for WSN/IoT data analysis and penetration testing by concentrating on implementing various intrusions using signal processing block based software like Simulink or GNU Radio. A comparison with traditional WSN packet sniffing/debugging tools is provided and the main security vulnerabilities of existing WSNs are surveyed by adopting the ZigBee protocol. An extension to WSN security analysis and testing is established by utilizing low-cost SDRs and specifying the ease of implementing various analysis techniques even when certain equipment, such as anechoic chambers, are unavailable. Stemming from previous simulations, the benefits of obtaining the in-phase and quadrature-phase samples, both with and without external interference, is also discussed.**

***Index Terms*—Data, IEEE 802.15.4, Interference, Intrusion, IoT, SDR, Security, Sniffer, WSN and ZigBee.**

I. Introduction

Wireless Sensor Network (WSN) applications are evolving, based on over a decade of research and development, into integral components of modern society and various safety-critical applications exist [1]. This evolution inherently produces new challenges in terms of privacy and safety, where securing the communication link against spectral coexistence and malicious intrusions, while identifying the threat, becomes imperative. Examples of the diverse application field include space-based WSNs [2], remote patient monitoring [3] and implementing the sensor to access point communication in the Internet of Things (IoT) [4], which reduces the computational load on, typically, resource constrained commercial off the shelf (COTS) IoT devices. However, securing these applications is challenging as these constrained devices hinder the use of computationally intensive security protocols, and WSNs in use have very similar physical (PHY) and medium access control (MAC) layer designs, which are typically based on open-source standards [1]. Hence, security vulnerabilities are identifiable and network compromise, whether malicious or unintentional, is achievable. This, combined with predictions that over 20 billion connected devices are forecast by 2022, emphasizes the need for secure networks and identifying why communications links fail, where possible.

Software defined radios (SDRs) allow for the rapid prototyping and field testing of new protocols and algorithms. New digital communication technologies are looking towards SDRs as a platform to expand the application space and improve communications [5]. In a SDR, the core logic of a required protocol is implemented in software, thereby enabling both changes easily and the rapid field-testing of algorithms for various inputs and approaches. Hence, SDRs can be beneficial for testing WSN responses to malicious attacks and enabling debugging processes, which assist developing penetration tests and security algorithms. This paper identifies how SDRs grant access to and produce data which can, potentially, enhance security in WSN/IoT devices by classifying link failures.

This study demonstrates the advantages of utilizing SDRs, and available software packages, as WSN signal analysis and penetration testing tools. Emerging from previous work [6], [7], extracting received samples in coexistence with both unintentional and malicious interference has potential security benefits and SDRs enable access to this data. The SDR architecture, which can be manipulated and provide local and remote access, is specified for producing various jamming scenarios in a real WSN/IoT testbed by utilizing Simulink, GNU Radio and a Raspberry Pi/Python combination. Throughout this paper, the main SDRs employed are low-cost COTS devices with available plug-ins for either Simulink, GNU Radio or Python. SDRs are compared to the traditional WSN debugging platform, the so-called packet sniffer, which provides network traffic data and statistics including the received signal strength indicator (RSSI) and link quality indicator (LQI). These metrics identify the channels with high levels of interference and are most likely to incur packet errors, but the cause is not specified, while SDRs provide access to raw data from the channel for post-processing and analysis.

The remainder of this paper is organized as follows. Section II summarizes related work in this area. Section III describes WSNs and their involvement in IoT applications. Section IV discusses available SDR hardware and compares the output to conventional packet sniffers. Section V specifies example implementations for deploying SDRs as WSN inspection devices and penetration testers, while the value of accessing in-phase (I) and quadrature-phase (Q) samples is also described. Section VI concludes this paper and provides future work.

II. Related Work

Using SDRs in WSN protocol and application development is not an entirely original concept, but one which is advancing to include several use cases due to their versatility and performance. Initially using a packet sniffer to aid WSN research is evident in [8], where a packet sniffer is used as a network traffic tool to debug and develop code for a ZigBee application involving small robots and tens of modules. The packet sniffer identifies which modules required retries and what exception was being transmitted by the ZigBee stack. In [9], the limitations of using a traditional packet sniffer as a debugger include the difficulty of scaling as the number of sniffed packets becomes large. In [10], a packet sniffer exploited a ZigBee network by sniffing the security keys on a node join process, which again identifies the usefulness of packet sniffers for protocol development. SDR use cases include software defined networks [11] aimed at WSNs for industrial IoT and a multi-channel packet sniffer [5] developed using a universal software radio peripheral (USRP) and the Wireshark software package. Some other typical packet analyzers include Texas Instruments smart packet sniffer, Perytons protocol analyzer and Ubiqua protocol analyzer. The paper produced a 5 channel sniffer, produced RSSI, LQI and packet error statistics and also discussed the concept of developing a protocol that can dynamically adapt to its environment. Using a SDR to analyze wireless signals for the presence of interference was discussed in [12], where a GPS jamming detector was developed using a RTL-SDR dongle and analysis of received samples. The work in this paper differs from the literature, as focus is applied to using low-cost SDRs to retrieve and analyze real-world data signals for the presence of interference, both malicious and otherwise. No packet analysis program is used and using received I/Q samples is motivated from prior interference detection investigations [6], [7].

III. WSNs & IoT: Discussion

Here, the IEEE 802.15.4 based low rate wireless personal area network (LR-WPAN) protocol, ZigBee, is adopted. It is the de-facto standard for WSNs, due to almost all available commercial and research sensor nodes being equipped with ZigBee transceiver chips [13]. ZigBee enabled devices are essential for the all-inclusive IoT communication architecture, shown in Fig. 1, as, typically, the sensing/actuating devices (or "things") use a LR-WPAN to communicate with other equipment and the internet access point. This utilization reduces node firmware complexity, supports a higher number of connected devices and a longer range (compared to wireless local area networks), while the available mesh topology automatically configures routing between devices. A successful IoT deployment requires reliable sensor and control data and, so, each WSN link needs to be secure and maintain uninterrupted, safe and non-malicious operation.

TABLE I
IEEE 802.15.4 (ZigBee) PHY Parameters

Parameter:		**2.4 GHz PHY Value:**	
Number of Channels		16	
Channel Width	Spacing	$2\ MHz$	$5\ MHz$
Data Rate	Chip Rate	$250\ kb/s$	$2\ Mchips/s$
Symbol Rate		$62.5\ ksymbols/s$	
Modulation Scheme		OQPSK	
Pulse Shaping		Half Sine/Normal Raised Cosine	
Byte Spreading		DSSS	
Maximum Packet Length		133 bytes	

ZigBee adopts the PHY (Table I) and MAC layers from the IEEE 802.15.4 standard and, here, the $2.4\ GHz$ unlicensed industrial, scientific and medical (ISM) radio frequency (RF) band is chosen. ISM band ZigBee transmissions can be visualized by a Tektronix real-time spectrum analyzer (RTSA) and associated digital phosphor technology (DPX), which performs hardware digital signal processing and rasterizing of samples into pixel information. Various signals, including Bluetooth, other LR-WPANs and WiFi, coexist in the ISM band and an example of this coexistence issue is provided in Fig. 2. Each operating center frequency is supplied in (1), where F_c is the center frequency and i is the channel number. ZigBee's operating topology is either star, mesh or peer-to-peer and, in each case, is self-organizing, self-repairing and can exploit clustering approaches [14]. Direct sequence spread spectrum (DSSS) splits every byte into two 4-bit symbols, which are each spread to a predefined 32-bit pseudo-noise (PN) sequence. Transmitted signals are modulated using offset quadrature-phase shift keying (OQPSK), which mutually offsets the I and Q components by half a symbol duration, and pulse shaped using the raised cosine or half-sine method, which ideally achieves the desirable property of zero inter-symbol-interference at the maximum effect points. Carrier sense multiple access with collision avoidance (CSMA/CA) accesses the channel and a clear channel assessment determines whether the channel is free or busy, prior to transmission.

$$F_c = 2405 + 5(i-11)MHz,\ for\ i = 11, 12, ...26 \quad (1)$$

Typical WSN devices have low processing power, memory and speed and operate on a finite energy source which, when combined with cost, impedes the use of conventional security protocols. However, to maintain safety and data privacy, security is a necessity and is required across a diverse range of WSN applications deployed in different operating environments. Generally, WSN security can be described in terms of requirements, vulnerabilities, attacks and defenses [15]. This study focuses on the vulnerabilities and attacks, which, along with deployments in hostile environments and/or in safety critical applications, intensifies the need for se-

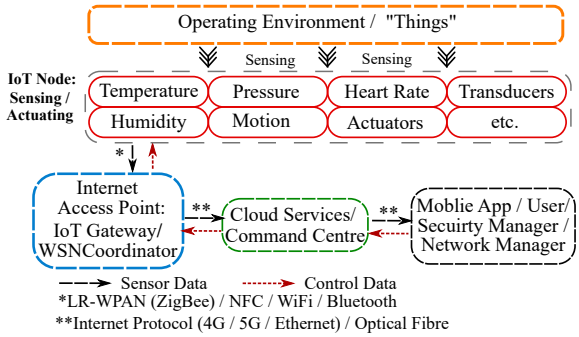


Fig. 1. IoT Architecture showing the use of ZigBee

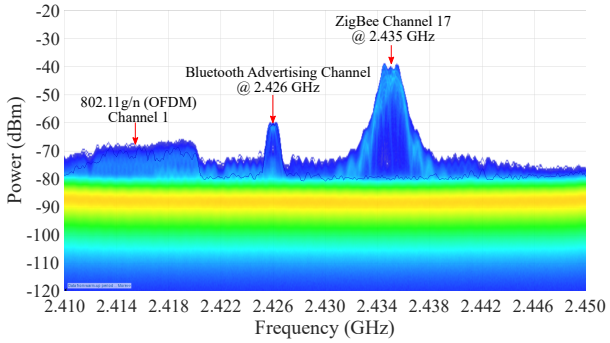


Fig. 2. ZigBee signal, visualized using a Tektronix RTSA and its DPX software, in the ISM band @ 2.435 GHz with coexisting signals

curity. Guaranteeing data confidentiality and authenticity in WSNs can be difficult due to known vulnerabilities [1], [16], including the open interface of the wireless channel and (unavoidably) publicly known WSN protocols. Frequently, device deployments are in hostile or remote environments, where continued surveillance is difficult to guarantee and nodes can be physically available to potential attackers. In addition, WSN attack types are numerous and can occur across the entire communication protocol stack, where applications and sensitive data incentivize attacks, which can vary from denial of service (DoS) attacks, that can corrupt all packets, to privacy attacks, that can aim to seize sensitive data. Here, jamming attacks are employed to investigate the usefulness of SDRs as WSN penetration testers and analysis tools.

IV. Hardware Investigation & Analysis

Analyzing WSN operation in real-time, while simultaneously providing metrics describing the wireless channel and the packets in transit, can be beneficial. This concept is illustrated in previous work aiming to develop interference detection systems (IDS) using received I/Q samples [6], [7], which require hardware to extend simulated approaches to real wireless environments. Initially, a comparison between a traditional packet sniffer and a modern low-cost SDR is provided to demonstrate the advantages of using SDRs and how access to I/Q samples is granted. Here, the Texas Instruments CC2531 is the chosen packet sniffer, while the Analog Pluto and LimeSDR Mini are the (relatively) low-cost SDRs. These SDRs were validated for use in WSNs in [17], in which the Pluto was used to transmit matched signal interference [18] using Simulink and the LimeSDR Mini acted as a WSN transceiver and was shown to operate remotely on Raspberry Pi 3 B+ by utilizing GNU Radio. Table II describes the wide range of functionality of various SDRs, whose frequency range reaches the ISM band, and includes the relatively expensive Ettus USRPs. However, the desired SDR WSN analyzer and penetration tester should be relatively low-cost and, so, as multiple lower-cost SDRs can be acquired for the price of a single USRP, Ettus products were deemed out of scope here.

A. TI CC2531EMK Packet Sniffer

For traditional packet sniffers, the sniffed packet structure is significant. Results are obtained based on the sniffed signal's frame and can be used to debug networks by determining, for example, link quality and retransmissions. Fig. 3 specifies a simplified expanded ZigBee packet structure showing the different frame segments. TI's CC2531 USB dongle employs TI's Packet Sniffer software to both capture and decode ZigBee packets. The sniffer provides information as displayed in Fig. 3 where, for example, the packet type (data, acknowledgment, etc.) is found in the MAC's frame control field while the source and destination addresses are in the address information. The sniffer is capable of monitoring one channel at a time and can report the channel's RSSI and LQI metrics for each received packet. RSSI is a measure of the energy contained in a received signal and is reported in dBm, while the LQI provides the error in the received signal modulation. A Raspberry Pi, or Linux system, enables limited operation of the dongle but still supplies a useful set of information based on the frames in Fig. 3. Notably, the packet sniffer does not provide the received samples, nor has it any functionality for transmitting a packet, thus, an extra transmitting device is required. As a result, these

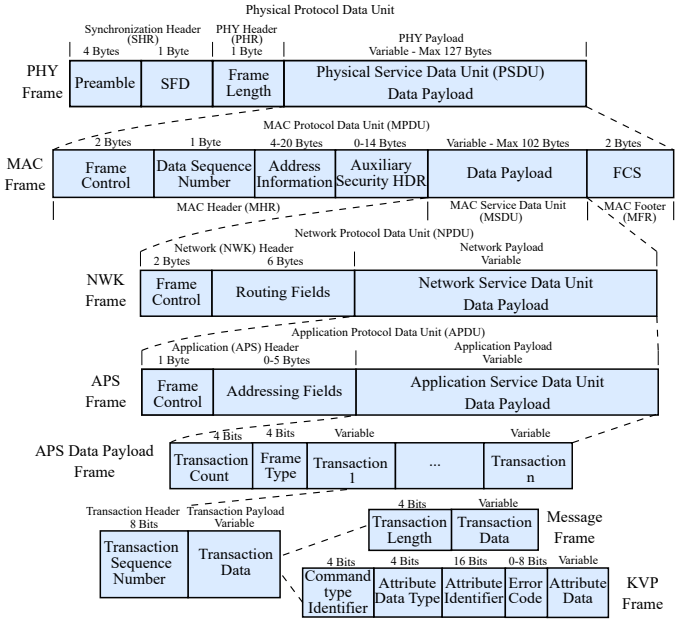


Fig. 3. A simplified ZigBee frame structure visualizing the typical data output of a traditional packet sniffer by specifying frame specific bytes

TABLE II
SDR SPECIFICATIONS

SDR	Interface	Frequency Range	RF Bandwidth	ADC Resolution	Mode	TX/RX Channels	Approx. Price
LimeSDR	USB 3.0	$100KHz \rightarrow 3.8GHz$	$61.44MHz$	12 bit I/Q	Full Duplex	2/2	$299
LimeSDR Mini	USB 3.0	$10MHz \rightarrow 3.5GHz$	$30.72MHz$	12 bit I/Q	Full Duplex	1/1	$159
Analog Pluto	USB 2.0	$325MHz \rightarrow 3.8GHz$	$20MHz$	12 bit I/Q	Full Duplex	1/1	$149
HackRF	USB 2.0	$1MHz \rightarrow 6GHz$	$20MHz$	8 bit I/Q	Half Duplex	1/1	$299
BladeRF	USB 3.0	$300MHz \rightarrow 3.8GHz$	$40MHz$	12 bit I/Q	Full Duplex	1/1	$420
FreeSRP	USB 3.0	$70MHz \rightarrow 6GHz$	$61.44MHz$	12 bit I/Q	Full Duplex	1/1	$420
Ettus B200	USB 3.0	$70MHz \rightarrow 6GHZ$	$61.44MHz$	12 bit I/Q	Full Duplex	1/1	$796
Ettus B210	USB 3.0	$70MHz \rightarrow 6GHZ$	$61.44MHz$	12 bit I/Q	Full Duplex	2/2	$1119

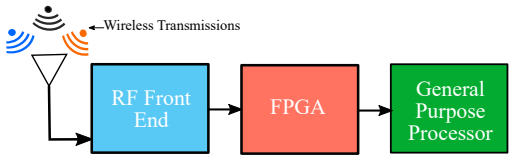


Fig. 4. Simplified depiction of a typical SDR topology

device types are commonly used as WSN debugging tools but not for received signal feature engineering or penetration testing. This principle deters using packet sniffers for WSN analysis as it only provides packet information on reception, while SDRs have the ability to provide I/Q samples for off-line analysis and protocol development.

B. SDR Hardware

SDRs are reconfigurable radio systems whose characteristics are partially or fully defined via software or firmware [19]. A typical simplified SDR topology is provided in Fig. 4, where the main components are the antenna, RF front-end and processing unit. A SDR interacts with the wireless environment using a hardware peripheral, whose capabilities characterize transceiver operation, and the performance of the software component depends on the proficiency of the RF front-end. Received analog RF signals are converted into a digital sequence, which depends on the available bandwidth and sampling rate in use. Hence, it is necessary to use SDRs which have the appropriate hardware for analyzing the chosen RF signals. This study focuses on developing an IDS in the ISM RF band, which relies on receiving the I/Q samples of a $2\ MHz$ wide signal and implementing penetration tests.

Here, the LimeSDR Mini and Analog Pluto SDRs are employed as WSN analysis and testing tools. The manufacturers describe the Lime Microsystems LMS7002M based LimeSDR Mini ($\approx$ \$159) as "the perfect way to start experimenting with and building your own wireless networks, protocols, and testers", which requires the LimeSuite package to operate as the WSN analyzer and penetration tester. The Pluto SDR ($\approx$ \$149) is based on the Analog Devices AD9363 transceiver and requires the "libiio" package. Both SDRs can be controlled by using SDR software including GNU Radio, Pothos Flow, CubicSDR, GQRX SDR, SDRConsole, etc., either through signal processing blocks or graphical user interfaces, where the SDRs can be, frequently, exploited as spectrum analyzers. In addition, the Pluto can be controlled by Simulink, using the Communications Systems Toolbox add on, and Python (pyadi-iio) or C/C#/C++, using the "libiio" library.

The Pluto's Simulink/Matlab plug-in provides further benefits through the available toolboxes, for example, Signal Processing and Communications. These toolboxes, along with the transmitter and receiver Simulink blocks, provide efficient analysis methods spanning different modulation methods, filters, mixers, etc. Both of these SDRs can also exploit the available GNU Radio plug-in, to provide remote access and deployment using the Linux based Raspberry Pi [17]. For lightweight Pluto operation, the Python3 "pyadi-iio" library is used. Typically, the chosen RF antenna is a critical element of the analysis tool and/or tester design. Therefore, a ZigBee Siretta stubby antenna, designed for use in the $2.4 \rightarrow 2.5GHz$ range, was selected, whose specifications include a $2\ dBi$ gain, vertical polarization, a maximum VSWR of 2.0 and an input impedance of 50 Ω. The Pluto SDR, LimeSDR Mini and RTSA, which captured and visualized the ZigBee signal in Fig. 2, all employ this Siretta $2.4 \rightarrow 2.5GHz$ antenna.

By utilizing these devices, raw received I/Q data can be analyzed to gain an understanding of why links fail and what is causing the interference levels to rise. This concept has been proven in ZigBee simulations using a support vector machine in [6] and a Random Forest decision tree detection algorithm in [7]. Notably, the SDRs can receive I/Q samples even when ZigBee packets are erroneous and the packet error rate (PER) is close to 1, while many packet sniffers need to be able to synchronize to the packet preamble and identify the start frame delimiter. This paper uses this concept to outline why SDRs are effective in terms of WSN signal/samples analysis and for a subsection of IoT penetration testing, by demonstrating how simulation-based machine learning models for interference detection can be initially adapted to a wireless environment under legitimate and attack situations.

V. SDR IMPLEMENTATION: RESULTS

For this study, a live WSN/IoT testbed is required to test the SDR implementation. The ZigBee testbed designed in [17], which incorporated 6 XBee nodes connected to 6 Raspberry Pi devices and transmitted user-specific strings, was modified to transmit environmentally sensed data, including temperature, humidity and pressure. The Raspberry Pi SenseHat sensor was utilized as it easily connects to available GPIO pins and can imitate an actuator using the LED matrix. Testbed validation occurred over multiple tests, where five SenseHats collected environmental data and XBee devices transmitted the data to a central coordinator. Operation was controlled using various types of Raspberry Pis and the results are provided in Table

TABLE III
ZIGBEE TESTBED VALIDATION: SENSEHAT DATA

Node	One	Two	Three	Five	Six
Validation Test One - Node Four as Coordinator(Receiver)					
Approx. Operating Time (Hours)	64	64	64	64	64
Data Packets Transmitted	2042	2042	2006	2006	2042
Data Packets Received	2042	2042	2006	2006	2042
Validation Test Two - Node Four as Coordinator(Receiver)					
Approx. Operating Time (Hours)	24	24	24	24	24
Data Packets Transmitted	749	749	749	749	749
Data Packets Received	749	749	749	749	749

III. To enable IoT abilities, a WiFi-enabled coordinator can use available tools, for example, DropBox Uploader exploiting a DropBox API application, to upload received data to the internet for remote analysis. Data were verified as ZigBee signals by sniffing the channel using the TI packet sniffer and for zero packet loss by tracking all transmitted and received data packets. Thus, the sensor level communication of the IoT architecture, Fig. 1, is implemented and forms the foundation of the WSN analysis and penetration testing.

A. Penetration Tester

A penetration test is performed to evaluate the security of a system and, in terms of WSNs and IoT, specific penetration tests could involve assessing responses to external interference. Thus, a SDR having an RF front-end capable of transmitting in the ISM band can be used as an interference response tester. In this study, the penetration testing approach uses Simulink signal processing blocks, where the attack style can be designed and implemented by exploiting various toolboxes. An example is provided in Fig. 5, in which the matched protocol attack approach [18] is demonstrated. A dummy ZigBee PHY payload is populated, along with the required preambles, and spread according to the 32-bit PN sequences. OQPSK and raised cosine pulse shaping are applied to the chips and transmitted in the ISM band using the Pluto SDR. This produces "ZigBee" interference, which matches expected spectral images (Fig. 2) but, due to limitations of the transmit power ($\approx 7dBm$), a power amplifier is required to produce significant interference levels. The Analog Devices CN0417 provides an additional 20dB of gain and all ports are DC blocked and matched to 50 Ω. Similarly, GNURadio can be leveraged to implement this form of external interference.

Typical designs for penetration test orientations are shown in Fig. 6, including wireless and wired approaches as, typically, emitting wireless jamming signals is prohibited, unless in an anechoic chamber. SDRs can be connected through a DC block and power combiner to produce I/Q samples containing various interference signals, without affecting neighboring networks. This approach differs from typical signal generator [12] methods as a lower cost, software configurable and open-source technique is produced, which has remote deployment potential. However, a 50 Ω termination must be connected to the power splitter, either the power amplifier or a specific termination block. In addition, Python has a Pluto library, allowing Python's multitude of data science approaches to be

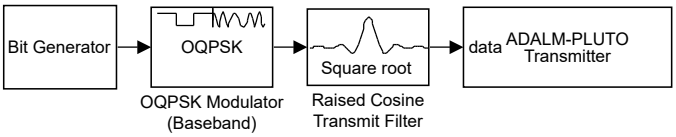


Fig. 5. An example Simulink OQPSK Pluto SDR transmission

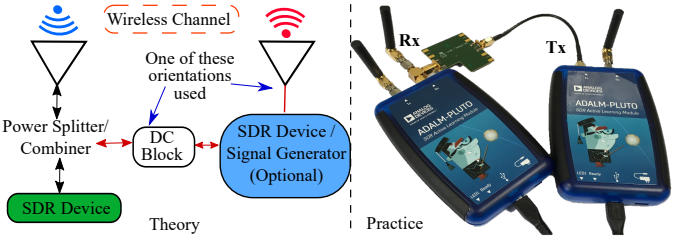
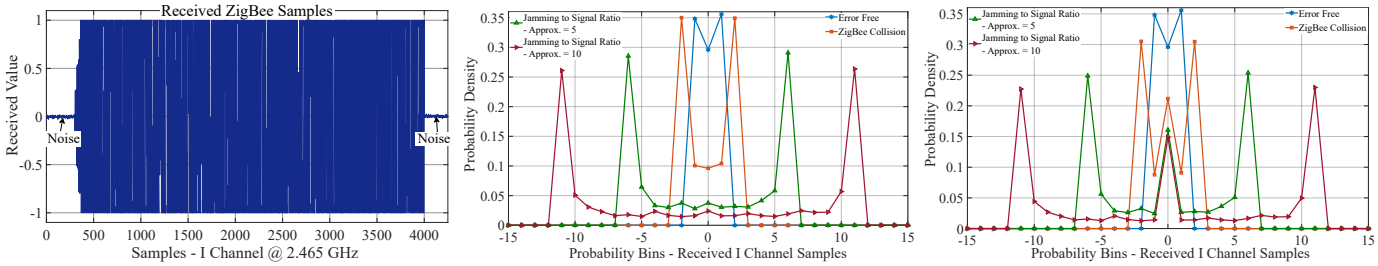


Fig. 6. SDR WSN/IoT analysis and testing approach, Theory and Practice

applied to received data points, emphasizing using SDRs as interference transmitters and received response analyzers.

B. WSN I/Q Sample Analysis

For WSN analysis, a SDR can simply receive on the required channel and acquire I/Q samples for off-line examination. Fig. 7(a) demonstrates this process by providing received ZigBee samples, on channel 23, using a $4MHz$ sampling rate, resulting in four samples per chip. The motivation for access to I/Q samples was specified in [6] and [7] using ZigBee Matlab simulations, where various forms of interference were applied to ZigBee transmissions. A feature set, extracted from the probability distribution function (PDF) and statistical analysis of the samples, was used in a support vector machine and a Random Forest approach to detect interference in received ZigBee I/Q samples, which are, generally, not publicly available on commercial WSN nodes or by using traditional sniffers. Therefore, SDRs were the key element in translating and, potentially, validating the simulated approach in a real wireless application, as samples can be received even when the channel becomes noisy and packets are lost or erroneous, since no preamble is required, as reception can be linked to known packet transmission times or periods. In this manner, erroneous, error-free and interference data are collected for feature extraction and analysis. Once analyzed, results can be implemented as part of the designed machine learning detection approach to highlight why a link has failed and packets have been lost. Hence, WSN security, in terms of attack detection, is enhanced and Fig. 7(b) (ZigBee samples only) and Fig. 7(c) (ZigBee samples and noise tails) signify that live data adheres to the simulated approach as the PDF becomes more bimodal as interference increases. However, the choice of samples is vital, as the PDF differs when extra samples are used, emphasizing the need for accurate sample detection or identification of specific windows. Differentiation exists from comparable literature [5], as no additional software program is required, since all analysis is done on the sample level using low-cost SDRs. In addition, this study focuses on sample variations under interference, while [5] focuses on packet capturing for protocol development. Thus, ZigBee



(a) (b) (c)

Fig. 7. Provisional results of using the Pluto SDR and a 4 MHz sampling rate, to emphasize access granted to the I/Q samples, motivated by the simulation work in [6] and [7]. (a) Example of non-interfered ZigBee I samples on the 2.465 GHz channel. (b) Example PDF of Pluto received ZigBee I samples on channel 23, confirming simulated results. (c) Example PDF of Pluto received ZigBee I samples on channel 23, showing the effect of additional noise samples.

simulations motivated using received I/Q samples to detect interference and SDRs facilitate testing, sample analysis and designed simulation-based methodology validation.

VI. Conclusion

This investigation adopted previous simulation based IDS designs to motivate the exploration of accessing I/Q samples and facilitating interference situations in real WSN environments. SDRs were established as both WSN/IoT analysis tools and penetration testers concerning external interference scenarios. Available low-cost SDRs and suitable software packages were identified, while the benefits of Simulink and Python were specified. The link between WSNs and the IoT was illustrated and used to update and validate a ZigBee testbed, to include SenseHat sensors and an IoT connection using DropBox. By exploiting this testbed and available Simulink plug-ins, the Analog Pluto SDR provided I/Q samples for analysis, even when packets were erroneous, and produced matched protocol interference. Receiving samples in the presence of strong channel interference is the key identified advantage that SDRs provide over traditional packet sniffers. A testing method, which embraces this concept and connects multiple SDRs, was developed to test the effects of interference with/without interfering with other ISM band services in use. Signal processing blocks and coding libraries allow various attacks to be implemented, which can produce specific datasets to expand previous simulation work in detecting WSN interference. Future work includes assembling datasets of received I/Q samples for specific signals (Noise, WiFi, ZigBee, etc.) and for WSN signals under distinct interference patterns. The data will help to evaluate if the extracted simulation features are applicable to real wireless signals. In essence, this SDR study was crucial for developing the WSN IDS data strategy.

Acknowledgment

This work has been jointly funded by the Irish Research Council (IRC) and Raytheon Technologies Research Center, Ireland, under the post-graduate Enterprise Partnership Scheme 2016, award number EPSPG/2016/66.

References

[1] G. D. O'Mahony, P. J. Harris, and C. C. Murphy, "Investigating the Prevalent Security Techniques in Wireless Sensor Network Protocols," *Proc. 30th IEEE Irish Signals Syst. Conf. (ISSC)*, 2019, pp. 1–6.

[2] T. Vladimirova, C. P. Bridges, J. R. Paul, S. A. Malik, and M. N. Sweeting, "Space-based wireless sensor networks: Design issues," *IEEE Aerosp. Conf.*, pp. 1–14, 2010.

[3] S. Tennina, M. Santos, A. Mesodiakaki, et al., "WSN4QoL: WSNs for remote patient monitoring in e-Health applications," in *2016 IEEE Int. Conf. Commun.*, May 2016, pp. 1–6.

[4] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things : A survey," *Comput. Networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[5] L. Choong, "Multi-Channel IEEE 802.15.4 Packet Capture Using Software Defined Radio," Ph.D. dissertation, UCLA, 2009.

[6] G. D. O'Mahony, P. J. Harris, and C. C. Murphy, "Identifying Distinct Features based on Received Samples for Interference Detection in Wireless Sensor Network Edge Devices," in *2020 Wirel. Telecommun. Symp. (WTS), Washingt. DC, Virtual*, 2020, pp. 1–7.

[7] G. D. O'Mahony, P. J. Harris, and C. C. Murphy, "Detecting Interference in Wireless Sensor Network Received Samples: A Machine Learning Approach," in *IEEE Virtual World Forum on Internet of Things (WF-IoT V2020)*, Virtual, 2020, pp. 1–6.

[8] R. Fitch and R. Lal, "Experiments with a ZigBee wireless communication system for self-reconfiguring modular robots," *IEEE Int. Conf. Robot. Autom.*, pp. 1947–1952, 2009.

[9] A. Wheeler, "Commercial applications of wireless sensor networks using ZigBee," *IEEE Commun. Mag.*, vol. 45, no. 4, pp. 70–77, 2007.

[10] T. Zillner, "ZigBee Exploited The good, the bad and the ugly," *Black Hat USA*, 2015. [Online]. Available: http://www.sicherheitsforschung-magdeburg.

[11] Y. Duan, W. Li, X. Fu, Y. Luo, and L. Yang, "A methodology for reliability of WSN based on software defined network in adaptive industrial environment," *IEEE/CAA J. Autom. Sin.*, vol. 5, no. 1, pp. 74–82, Jan. 2018.

[12] G. D. O'Mahony, S. O'Mahony, J. T. Curran, and C. C. Murphy, "Developing a low-cost platform for GNSS interference detection," in *Eur. Navig. Conf.*, 2015, pp. 1–8.

[13] B. Stelte and G. D. Rodosek, "Thwarting attacks on ZigBee - Removal of the KillerBee stinger," in *Proc. 9th Int. Conf. Netw. Serv. Manag.*, 2013, pp. 219–226.

[14] I. Tomi and J. A. McCann, "A Survey of Potential Security Issues in Existing Wireless Sensor Network Protocols," *IEEE Internet Things J.*, vol. 4, no. 6, pp. 1910–1923, 2017.

[15] G. D. O'Mahony, J. T. Curran, P. J. Harris, and C. C. Murphy, "Interference and Intrusion in Wireless Sensor Networks," *in IEEE Aerosp. Electron. Syst. Mag.*, vol. 35, no. 2, pp. 4–16, 1 Feb. 2020.

[16] Y. Zhou, Y. Fang, and Y. Zhang, "Securing Wireless Sensor Networks: A Survey," *IEEE Commun. Surv.*, vol. 10, no. 3, pp. 6–28, 2008.

[17] G. D. O'Mahony, P. J. Harris, and C. C. Murphy, "Developing Low-Cost Testbeds for Enhancing Security Techniques in Wireless Sensor Network Protocols," in *2019 30th IEEE Irish Signals Syst. Conf. (ISSC)*, Maynooth, Ireland, 2019, pp. 1–6.

[18] ——, "Analyzing the Vulnerability of Wireless Sensor Networks to a Malicious Matched Protocol Attack," in *2018 Int. Carnahan Conf. Secur. Technol. (ICCST)*, Montreal, QC, 2018 pp. 1-5.

[19] J. T. Curran, C. Fernandez-Prades, A. Morrison, and M. Bavaro, "The Continued Evolution of Software-Defined Radio for GNSS," *GPS World*, no. 29, pp. 43–49, 2018.